Visualization of Decision Processes Using a Cognitive Architecture

Mark A. Livingston[a], Arthi Murugesan[b], Derek Brock[a], Wende K. Frost[a], and Dennis Perzanowski[a]

[a]Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC, USA;
[b]NRC Research Associateship Award at the Naval Research Laboratory, 4555 Overlook Ave SW, Washington, DC, USA

ABSTRACT

Cognitive architectures are computational theories of reasoning the human mind engages in as it processes facts and experiences. A cognitive architecture uses declarative and procedural knowledge to represent mental constructs that are involved in decision making. Employing a model of behavioral and perceptual constraints derived from a set of one or more scenarios, the architecture reasons about the most likely consequence(s) of a sequence of events. Reasoning of any complexity and depth involving computational processes, however, is often opaque and challenging to comprehend. Arguably, for decision makers who may need to evaluate or question the results of autonomous reasoning, it would be useful to be able to inspect the steps involved in an interactive, graphical format. When a chain of evidence and constraint-based decision points can be visualized, it becomes easier to explore both how and why a scenario of interest will likely unfold in a particular way.

In this paper, we present initial work on a scheme for visualizing autonomous reasoning that produces graphical representations of models run in the Polyscheme cognitive architecture. First, we give a brief overview of the architecture and note the key types of data that are critical for visual representations of cognitively-based, computational reasoning mechanisms. We propose an algorithm to generate visualizations of model-based reasoning, and discuss properties of our technique that pose challenges for our representation goals. Finally, we present example visualizations and simple interactions with the underlying chain of reasoning. We conclude with a summary of feedback solicited from domain experts and practitioners in the field of cognitive modeling.

Keywords: Software visualization, visual knowledge representation, graph/network data, visualization in social and information sciences

1. INTRODUCTION

The objective of this research is to investigate the advantages of a defeasible, cognitively-based, graphic representation of decision making at various levels in a command structure. Our intent is to promote informed situational understanding by giving decision-makers a way to visually explore the component elements of a given decision context. To facilitate this goal, we use a computational cognitive theory of human reasoning to identify the stages involved in a decision process and methodically construct an interactive display of this information that users can easily inspect and, if needed, alter to amend the premises upon which they are expected to act.

Decision-making in tactical operations involves the fusion of information that is typically distilled from a network of sources that can entail sensors, reports, and other types of systems and communications data. This information is then processed at intermediate levels where, in some cases, various stages of reasoning are likely to be automated in the near future. Ultimately, the result must be evaluated by a user or decision-maker with the authority to confirm or question and revise various aspects of the decision-making process. However, reasoning processes of any depth are often difficult for users to readily comprehend and can be even more opaque when computational systems are involved. Being able to see the constituents of a given process laid out in a path form, though, can help, particularly, if this provides an intuitive way for users to quickly examine key factors

Send correspondence to: Mark A. Livingston at mark.livingston@nrl.navy.mil, Telephone: 1 202 767 0380

and explore the efficacy of decision points involved in the result. By displaying each step in a manipulable, map-like format, it is thought that intermediate logic will be more conceptually concrete, which, in turn will allow individuals with operational authority to be more fully informed and more confidently prepared to act. As a case in point, automated reasoning by autonomous systems can present a unique problem for human decision-makers, since there is frequently no access to the underlying computations nor, more importantly, a high-level account of the system's performance parameters. Without an understanding of the latter, confidence in decisions and information derived from autonomous sources can only be achieved with experience over time and risks a range of unintended consequences. Decision-making interfaces, we therefore argue, should meet three conditions of adequacy: the interface should be cognitively-based, readily accessible, and easily understood.

Human decision-makers need cognitively-based explanation systems because they need to know the ramifications and repercussions of both human and autonomously generated information in order to weigh the options based on plausible evidence. Because of the typically fixed nature of current machine decision paradigms, contingencies that people need to be aware of are often hidden from the user. Augmenting automated decision-making with a human cognitive modeling approach that is designed to highlight and clarify situational logic and constraints will afford the decision-maker more confidence in accepting reasoning that involves autonomously made decisions. Secondly, information needed by decision-makers must be available at any point in the decision making process when the decision-maker requires it. Thus, ideally, any point in a decision network should be inspectable by the decision-maker. Thirdly, the presentation of information must be sufficient to allow the decision-maker to comprehend easily how a decision process was achieved. This calls for the use of straightforward language and a visual display of the overall decision path. To facilitate the user's understanding, it should also be possible to interactively explore and alter the underlying assumptions.

Several cognitive architectures have been designed to organize knowledge and produce intelligent behavior. Chief among these are ACT-R,[1] Soar,[2] and Polyscheme.[3] ACT-R and Soar both have mechanisms for resolving impasses and competing rules, but unlike Polyscheme, they do not reason in terms of alternative worlds. Furthermore, only the Polyscheme architecture incorporates Bayesian networks to model and reason over hypothetical circumstances. Even the simplest human actions can require the instantiation of several possible models to come to a resolution or decision. Polyscheme reasons over possible models and keeps track of decisions and other required data (e.g. sensory input from the physical world) during the evaluation process. Using the Polyscheme cognitive architecture coupled with a visualization component with the features outlined above, users will be able to easily inspect a model of the cognitive processes that have led to a particular decision.

Knowledge of how a system reaches its conclusions and decisions, what constraints may have to be broken, and how one model compares to other alternatives is invaluable information to a high-level decision-maker. Furthermore, the level of detail and the manner in which information is presented will have a direct impact on that individual's confidence in autonomous decision-making systems. Visualization algorithms for decision networks aim to convey complex relationships between multiple nodes, so that the information flow and decision-making process can be understood and, if necessary, validated or altered. In this research, we address the three conditions of adequacy for graphically representing decision processes elaborated above by developing cognitive models of specific scenarios and corresponding visualizations that allow this information to be effectively inspected by the user in an efficient manner.

2. RELATED WORK

We provide an overview of relevant and/or related work in two areas: the Polyscheme cognitive architecture and visualization techniques used for decision processes and related logical systems. One of our main goals is comparison of alternative structures, so we review some comparison techniques.

2.1 Cognitive Architectures

In this paper we develop decision models in the Polyscheme cognitive architecture. Polyscheme is an embodiment of the cognitive substrate hypothesis,[3] which postulates the existence of a core set of cognitive functions (a substrate) as the basis of most if not all of our high-level capabilities such as language, social behavior, and physical reasoning.[4] The substrate hypothesis is supported by research in artificial intelligence,[5] cognitive psychology,[6] linguistic semantics,[7] and neuroscience.[8] Although research on identifying all the elements of the substrate is ongoing, in its current form, the substrate is assumed to include both declarative and procedural knowledge of the concepts of time, category, identity, and uncertainty.

As mentioned earlier, Polyscheme is one possible implementation of the cognitive substrate. In the process of building a computational system that can function as the cognitive substrate, Polyscheme commits to additional computational and architectural principles beyond the substrate,[9] some of which are explained below.

Common Functions Principle Many different reasoning and problems solving strategies can be construed as a sequence of the same set of common functions such as forward inference, subgoaling, identity matching, and simulating alternate worlds.

Multiple Implementation Principle Each of the different common functions can be implemented using multiple algorithms and representations. For example, forward inference can be implemented by a production rule firing or a feed-forward neural network.

Focus of Attention At each computational cycle, all the different Polyscheme processes use their resources to compute only one piece of information, the focal "atom." By allowing different processes to execute independently but ultimately forcing them to converge on a consensus regarding the focal atom, Polyscheme achieves the desirable computational properties of both parallel processing and inter-algorithmic integration.

Polyscheme Specialists Specialized processors implement the common functions, using computational mechanisms that are different from specialist to specialist. Specialists implemented in Polyscheme include: a temporal perception specialist, a category specialist, an attribute specialist, an identity specialist, and a constraint specialist. In our decision models, we primarily use the temporal perception specialist and the constraint specialist.

2.1.1 Syntax of Polyscheme Elements

We briefly review the syntax of the Polyscheme elements used in the decision models we present below.

Atom The focal atom, which acts as the interlingua between specialists, is represented in a simple, familiar propositional formalism. `R(a1, a2, ..., aN, t, w)` is a generic atom that denotes a relation, `R`, that holds over N number of objects, `a1, a2, ..., aN`, during a temporal interval, `t`, in a "world," `w`. Forcing each atom to explicitly specify which world it belongs to aids users in representing and making inferences about alternate, possible, hypothetical, or counterfactual information.

Temporal Perception Specialist Perceptions and declarations can be asserted using the temporal perception specialist. This specialist can indicate its belief or doubt in an atom, called its *stance* on it, with a two-dimensional truth value expressed in a two-letter code. The first dimension is the degree of confidence it has in the proposition's truth, and the second is the degree of confidence that it is false. The five confidence levels `C, L, l, m`, and `n` indicate (respectively) that the truth or falsehood is certain, very likely, likely, maybe, or has no evidence. In `Color(rCar, Red, 2012, R) Cn`, `rCar` is perceived to be `red` during `2012` with certainty (with no evidence that it may be false).

Constraint Specialist The decision models discussed in this paper heavily rely on the constraint specialist. A Polyscheme constraint has three parts: the antecedent(s), the implication arrow, and the consequent(s). A conjunction of atoms, with the operator `^`, form the antecedents and consequents; note a negation of an atom, i.e. an atom prefaced with operator `-`, is also allowed. The constraint specialist is capable of interpreting the number on the implication arrow as either a weighted cost or a probability; in this paper, we limit our usage to the probabilistic approach.

Functionally, when all the antecedents on a constraint have been matched, the consequents are inferred. An object prefaced by a question mark, e.g. `?anyCar`, is used to represent a variable or a wild-card that can be matched by any possible value. Any unbound variable in the consequent results in a new object being created to fill that slot.

```
<constraint>
IsA(?passing, PassingEvent, E, ?w) ^
Occurs(?passing, ?t, ?w) ^
Agent(?passing, ?passingObject, E, ?w)
(.9)>
InMotion(?passingObject, ?t,?w)
</constraint>
```

The example constraint shown above captures the information that an object passing another is in motion at the time of the passing event, with a 90 percent chance.

Polyscheme output format Polyscheme performs inference by leveraging the world specialist and the focus of attention mechanism to guide a weighted search through possible alternate simulations of worlds. The output of Polyscheme, presented in an XML file, is a tree of worlds with intermediate nodes being worlds that have not yet been fully explored. Among the worlds that are the leaf nodes, the one with the highest probability is marked as the "best" world to indicate that it is the most likely outcome. Users are able to retrieve the atoms and their stances in the best world by traversing the tree from the best world to the root and collecting all the atoms during this process.

2.2 Visualizing Decision Processes

Few authors have directly addressed the issue of visualizing a decision process; thus, we begin our review of visualization methods with more general approaches to similar types of information. Classical approaches to visualization of logical inference include Venn diagrams, Euler circles, and Peirce's alpha system.[10] Such systems encapsulate consequences, but do not allow for decisions to be inserted into the inference process. Similarly, standard node-link diagrams have been applied to scientific inference.[11] Again, there is no allowance for judgments to be explored or questioned in standard approaches.

General Logic Diagrams (GLDs)[12] may be used to visually represent decision rules among many other logical constructs and functions. They use a table-based structure to capture the values of multiple variables; every cell represents a unique combination of values.

Slate[13] relies on natural language processing of a controlled form of English among other data entry formats. It uses node-link diagrams to represent deductive reasoning arguments based on known facts, then extends these to probabilistic inferences. Nodes are either facts, inferences, or inference types (with numerical indicators of certainty). Directed links proceed from source (evidence) to conclusion. In a controlled study, users solving with Slate spent more time solving an illusory inference problem, but were more accurate than users solving with paper and pencil.

Graph inscribed logic (Grailog)[14] diagrams are directed labeled graphs that represent simple semantic hierarchies. Instances of constants or variables lead to nodes, whereas logical predicates (binary or n-ary) yield edges of the graph. Hatching patterns distinguish between the two node types, while line styles depict types of logical relationships (positive, negative, disjunctive, subsumes, has instance, implies, equals). Conjunctions may be represented implicitly by co-occurrence on the top-level of the hierarchy or by explicit appearance in a compound node. Disjunction can also be represented by a compound node (using stipple effects). General classes use an oval-shaped node, instances use a rectangular node, and beliefs use an octagonal node (which may incorporate a relation among several instances). A logical relation is represented by a double arrow (similar to the mathematical symbol $\Rightarrow$) between the edges corresponding to the relations.

Defeasible logic is a rule-based approach that allows incomplete and inconsistent information. DR-DEVICE[15] implements this form of logic and three representations: tree, a more general directed node-link diagram, and text. A user evaluation found that a hybrid approach (using all three) was superior to the tree representation, but the authors found no difference between the hybrid and the graph representation.

2.3 Comparing Graphs and Trees

G-Pare*[16] offers three visual representations of graphs: tabular, matrix, and node-link diagrams. Nodes have attributes which represent confidence, divergence, and focus. Interactive operations, demonstrated in use cases, enable comparison of models at multiple scales. Nodes may be selected, with expansion to the ego-network (nodes immediately connected to current node), and filtered based on the attributes. Selected nodes are highlighted in all views (an implementation of the brushing interaction technique). The user may follow paths through the network using the ego-network expansion. This tool is envisioned to assist with correction of errors compared to a ground-truth model.

For a task of finding the least common ancestor in a hierarchy, traditional and orthogonal tree layouts were found to lead to faster performance than radial layouts.[17] Further, orientations of traditional and orthogonal trees that featured the root at either the top or bottom yielded faster performance than orientations that featured the root at the left or right. No differences were found in error rates. Trees also may be compared[18] through similarity measures of leaves, elements, and edges. These scores may then be used to color a similarity matrix or histogram to assist in comparison. More directly, the original tree structures may have the relevant elements highlighted for comparison. TreeVersity[19] enables comparison of hierarchical information through shape changes to nodes in traditional trees and partially-filled nodes in treemaps.

3. ALGORITHM

Polyscheme cognitive models have been previously used to make decisions.[20] However, the interactive aspect of the visualization tool requires user-friendly explanation of those decisions and exploration of alternate decisions as well. This interactive exploration is the key contribution of our work.

3.1 Customization of Polyscheme Input

We need two custom elements in our input to Polyscheme. First, we need to encode knowledge of the world in which a decision will be made. Second, we need a way to understand the propagation of this knowledge to likely outcomes.

3.1.1 Decisions

In order to make a well-informed decision, a user must be able to easily explore outcomes that correspond to different choices. We model the choices among decision points as if they are evidence that is already known. Polyscheme models typically represent this sort of prior knowledge or evidence as percepts in its temporal perception specialist. If, for instance, we choose to consider a scenario wherein a red car passes a black car, the corresponding elements in the decision would be modeled as follows.

```
<specialist name="perception.
AdvancedTemporalPerceptionSpecialist">
<step time="0">
IsA(passing123, PassingEvent, E, R) Cn
IsA(redCar, Car, E, R) Cn
IsA(blackCar, Car, E, R) Cn

Agent(passing123, redCar, E, R) Cn
Object(passing123, blackCar, E, R) Cn
</step>
</specialist>
```

Should the user need to examine a different case, wherein the black car passes the red car, the last two atoms could be changed to

```
Agent(passing123, blackCar, E, R) Cn
Object(passing123, redCar, E, R) Cn
```

As shown, by altering a few propositions and re-running the cognitive model, we are able to explore alternate situations.

3.1.2 Adding Tags to Decision Models

Polyscheme performs inference using a variant of the DPLL search algorithm named GenDPLL[21] which is an exact inference algorithm for probabilistic systems. Generating an explanation or justification of why the probabilistic system results in a particular answer is not as straight-forward as generating explanations for logic-based deduction systems, like Slate[13] or expert knowledge systems, like MYCIN.[22] Several researchers have attempted to address the problem of what is an appropriate explanation for a probabilistic system,[23] including the approach of listing the entire world with the highest probability.[5, 24] This paper does not attempt to identify ideal explanations of probabilistic inference in general terms, instead it limits itself to finding appropriate explanations for users of this specific decision system. One possible solution for generating a desired explanation is to somehow allow the user to specify which explanation is of interest.

As mentioned earlier, most of the information regarding the behavior associated with the decision is represented in the form of constraints. Furthermore, the modeler defines these constraints with diligence to provide the most accurate depiction of the user's real world situation as possible. Taking this into account, we have devised a scheme in which the modeler can aid the generation of explanation, by choosing or guessing which constraints the user would be most interested in as a form of explanation. Meta-data, which we call Tags, are appended to these constraints of interest. (In the next sub-section, we show how we retrieve the Tags and build an explanation.)

```
<constraint>
Near(?nearCity, ?targetCity, E, ?w) ^
IsA(?targetCity, targetLocation,
    ?tTarget, ?w) ^
IsA(?wpn, Weapon, E, ?w) ^
Location(?wpn, ?nearCity,
         ?tWeaponControl, ?w) ^
InControl(?wpn, gang, ?tWeaponControl, ?w)
(0.95)>
GangAttemptsToMove(?wpn, ?nearCity,
          ?targetCity, ?tWeaponControl, ?w)
</constraint>
```

The constraint shown above represents the expected behavior of a gang attempting to move weapons from nearby to a target location in a city. The Tag for such a constraint includes the following information.

1. *Tag ID* Each Tag has a Polyscheme generated unique identification number.
   `TagID(?tagNum, ?w)`
2. *Tag Type* Users can categorize constraints into different types for purposes of explanation, for example the constraint above is labeled to have Tag of type GangBehavior.
   `TAG(?tagNum, GangBehavior, ?w)`

3. *Antecedents* The number of antecedents in a constraint and the antecedent values are included in the Tag.
```
NumberOfTagAntecedents(?tagNum,5, ?w),
TagAntecedent1(?tagNum, Near,
 ?nearCity, ?targetCity, E, ?w)
```

4. *Consequents* Similarly, information about the consequent(s) is (are) available in the Tag.
```
NumberOfTagConsequents(?tagNum, 1, ?w) ,
TagConsequent1(?tagNum,
GangAttemptsToMove( ?wpn, ?nearCity,
    ?targetCity, ?tWeaponControl, ?w)
```

5. *Constraint Strength* This is the probabilistic strength of the constraint.
```
TagConstraintStrength(?tagNum, 0.95, ?w)
```

6. *Constraint Description* A user defined description of the semantics of this constraint.
```
TagConstraintDescription(?tagNum,
 Gang_attempts_to_move_
     weapons_to_target_city, ?w)
```

7. *Display Code* A short code used in the visualization of the tag.
```
TagDisplayCode(?tagNum, 3,?w)
```

3.2 Extracting a Reasoning Process from Polyscheme Output

Using the customized output of Polyscheme, we are able to determine nodes in a reasoning process and build a graph that represents this reasoning. We illustrate the algorithm using a simple example of automated reasoning about the following sentence (spoken by a human).

The red car just passed the black car.

The constraints in this world relevant to this exposition of the algorithm are the following.

1. If a passing event occurs, an agent precedes the event, and an object follows the event, then the agent is passing and the object is being passed.
2. If a passing event occurs, the passing agent is a car, and the passed object is a car, then there is some likelihood that the passed object may be in motion.

There are more constraints that Polyscheme would use to reason about this sentence; however, these two are sufficient to illustrate the algorithm.

The first step in the algorithm is to extract the nodes for the graph. When Polyscheme uses a constraint, we direct it to generate a Tag as described in the previous subsection. This will cause it to output three things: a statement of the existence of a Tag, a statement of the type for the Tag, and the strength of the Constraint this Tag places on the possible outcomes for the world. For the two constraints above, the following statements would be generated by Polyscheme.

```
Exists(tagNum251, E, R)
Exists(tagNum243, E, R)
TAG(tagNum252, Assignment, R)
TAG(tagNum243, State, R)
TagConstraintStrength(tagNum251, 1, #344)
TagConstraintStrength(tagNum243, 0.6, #344)
```

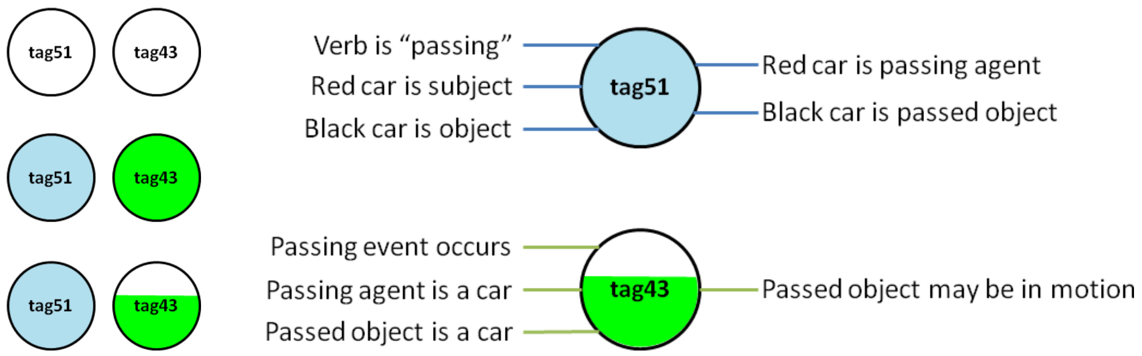

Figure 1. *Left*: The progression of creating a node begins with simple existence, continues on to coloring nodes by type (Assignment=blue, State=green), and then sets the fill of the node to the truth value (here, 1.0 and 0.6, respectively). *Right*: To create edges for nodes, we specify the antecedents and consequents of nodes, then look for shared statements. Here, Node 251 has consequents that the agent and object of the verb "passed" were both cars. These match two antecedents of Node 243, so there would be an edge created from Node 251 to Node 243.

Tag types are assigned in the input to Polyscheme; we could create any number of them that we like. The strength of a Tag is a truth value; in the case of uncertainty in the world, these values may take on any number in [0,1]. The final parameter in the constraint strength statement is the ID of the world in which these Tags had these strengths. As noted above, Polyscheme tries to reason through numerous alternatives; we use the "best" world as it defines it on the output. The existence of a Tag, its type, and its strength produce a progression of representations: a node, colored, and then (if appropriate) partly unfilled to represent a truth value of less than 1 (Figure 1). In each node, create a label from the instance ID of the constraint that Polyscheme generates.

The next step in the algorithm is the creation of edges between the nodes created in the first step. This is done in a two-stage process. In the first stage, all antecedents and all consequents of constraints that are output as Tags are listed on the respective nodes (Figure 1). The second stage matches consequents of one node with antecedents of another node. Such a match would indicate that an edge exists from the node with the matched consequent to the node with the matched antecedent. In this way, one can think of this process as discovering a dependency graph. By chaining together all of the constraints that were used in arriving at conclusions for the best world, we can understand the reasoning that led to those conclusions.

This simple example is missing one key element that we need in real scenarios: the ability for the user to make decisions. We denote decision points with a diamond; this will indicate to the user where options are available that may lead to alternative outcomes that could be more favorable.

3.3 Interactive Explorations

Such graphs of reasoning will be more helpful if the user can explore them to understand which constraints were engaged. We described three interactions with our graphs.

We use a "tool-tips" approach, enabling bubbles with constraint descriptions. These descriptions are also output by our customized Polyscheme as part of the Tag associated with each constraint that was used. These bubbles may also be applied to edges in the graph, with the description indicating the consequent-antecedent pair that created that edge. In the case of multiple pairs creating a single edge, all pairs are listed.

It seemed to us (and was confirmed by our domain experts, as discussed in Section 5) that an important interaction technique would be the tracing of a path backward through the graph to find the decisions that affected a particular outcome. This requires a simple recursive search through the ancestry of a node.

Of course, the most interesting interaction is to change a decision. We display alternatives of decisions as emerging from the decision node, drawn with a diamond shape. A decision that was made is displayed with a solid line; a decision that is not made is displayed with a dashed line. Clicking on this dashed line will change the decision. This (generally) will lead to an alternative set of outcomes, although not all internal nodes may disappear. Thus, new nodes will be created to lead to the new outcomes; edges that are no longer in use will also turn into dashed lines.

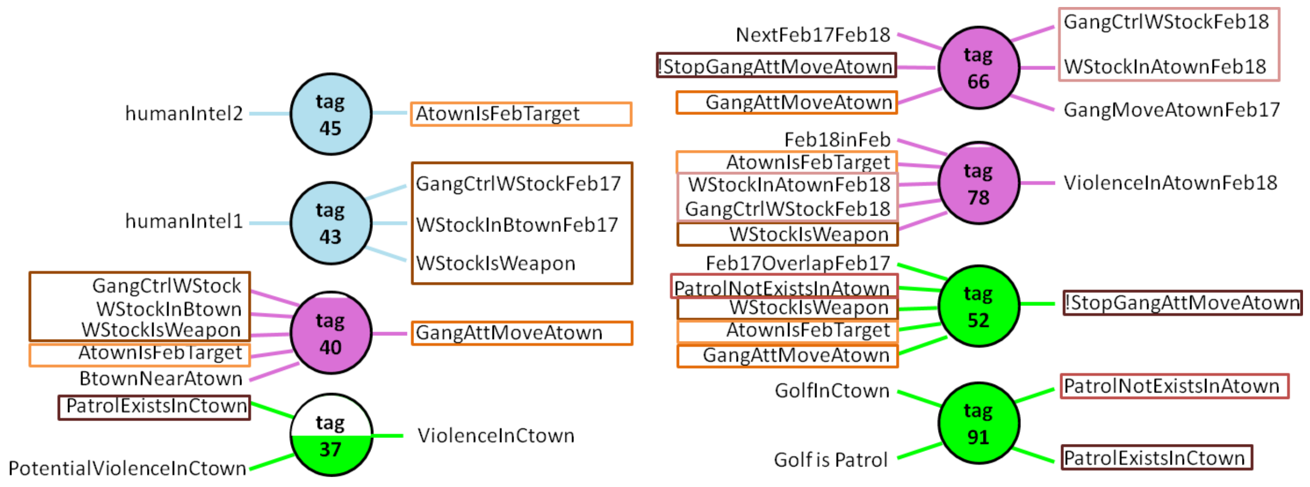

Figure 2. The results of our algorithm on a scenario that fits with our intended application. This shows the algorithm's progress through the matching of consequents and antecedents, just before the graph is built. The colored boxes are used for illustration of matches; none of this view is shown to users.

Images illustrating these interactions require a more complex scenario, and are thus deferred to the following section.

4. RESULTS

We present a scenario that fits with our intended application, allocation of resources by law enforcement. In this hypothetical example, the resource is a patrol, and the decision is where to assign that patrol. For the sake of space, we show the first four steps of the algorithm for extracting the graph, from creating nodes to the identification of matches between consequents and antecedents in Figure 2. Seven nodes are created, each from a different constraint. There are three node types: Intelligence (blue), Tactics (green), and outlaw Behaviors (lavender). Figure 2 also shows all the consequents and antecedents for each node. These labels are for exposition of the scenario; they reflect something that is generated from the arguments in each consequent and antecedent to uniquely identify and enable this matching.

We also created a special node of type Violence; this is the outcome that is most interesting for law enforcement in our scenario. Creating this special node type enables us to parse the consequents that do not match with any antecedents and thus appear on nodes that are outcomes. This node is similarly filled according to the truth value assigned; we gave it a special shape reminiscent of its meaning. The node definitions and properties along with the edges and their properties are translated into an input file for Graphviz,[25] which then draws the graph. We do a small amount of post-processing on the graph image, replacing nodes that we wish to have partly filled with a partial-fill rendering (a feature not available in Graphviz).

From the intermediate stage, Graphviz creates the graph for the computed best world (Figure 3). The graph shows three decisions that can be made. The first two are whether to believe the intelligence reports received; the third is where to place the patrol unit (the dashed arrow at the bottom left of the graph). The conclusion reached is that Violence will occur in Atown with a truth value of 0.94; one may conceive of this as the probability that Violence will occur.

We next describe the three interactions with the graphs. First, is the simple querying of information about a node. Using simple bubbles, a user can select a node and get a description of the constraint that was fired to create that node. These descriptions are embedded with the constraint as part of the Tag. This operation may be applied to a single node or applied recursively, as illustrated in Figure 4.

We next illustrate the changing of a decision, which in Polyscheme's parlance, implies a change of the input percept that encapsulates that fact about the world. When the user selects the decision to send the patrol to the other location, three new nodes appear in the graph (Figure 5). The first is trivial; it merely states the existence

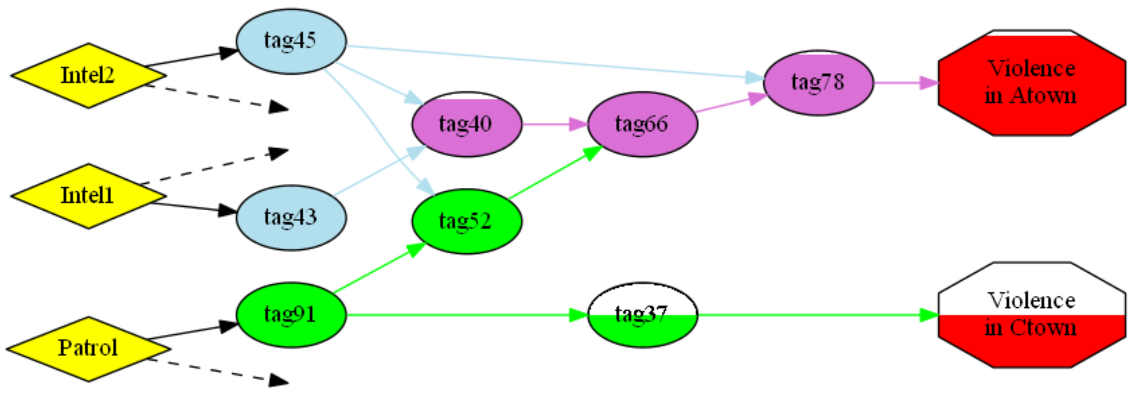

Figure 3. The final graph for the world computed from the input constraints. Three decision points (yellow diamonds) appear on the left; the conclusion (red "explosion" at right) is that Violence will occur in Atown with a truth value of 0.94. As per the discussion of Figure 2, Intelligence nodes are blue, Tactics nodes are green, and outlaw Behavior nodes are lavender.

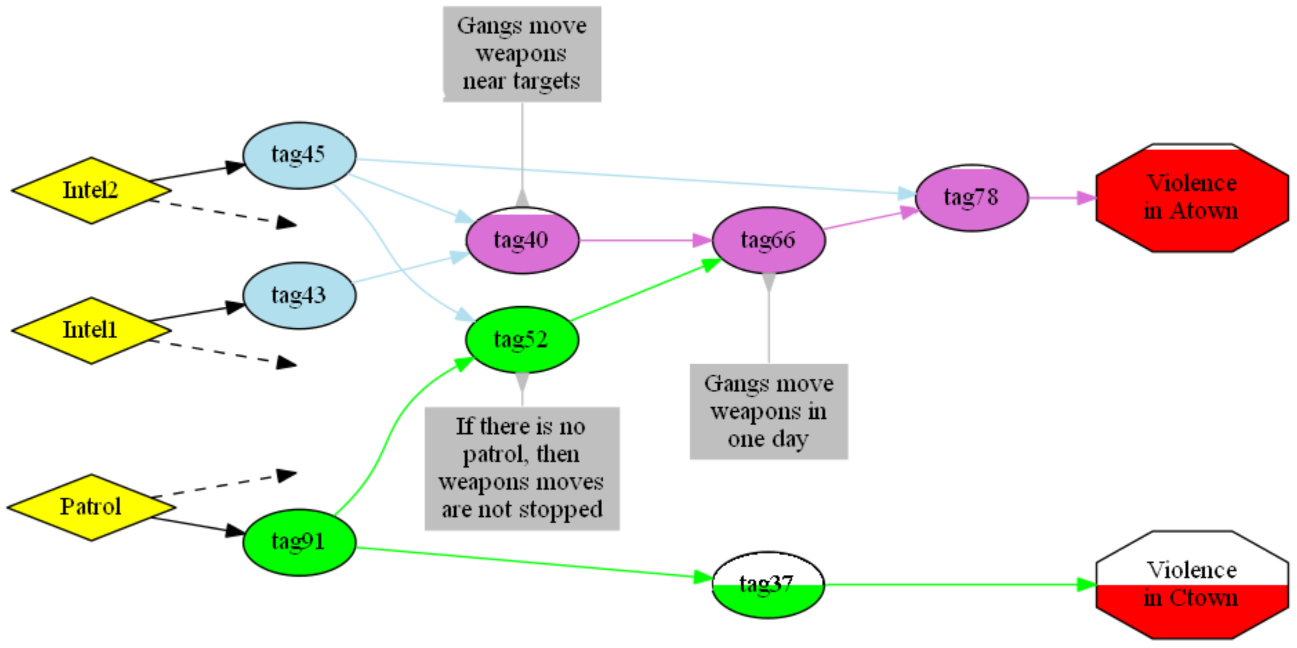
Gangs move weapons near targets
Intel2
Intel1
Patrol
tag45
tag43
tag91
tag40
tag52
tag66
tag37
tag78
If there is no patrol, then weapons moves are not stopped
Gangs move weapons in one day
Violence in Atown
Violence in Ctown

Figure 4. A simple interactive exploration of the graph enables the user to read the constraint descriptions associated with each node in the graph. Here, a node is selected, and the operation is applied to it. The operation is also applied through a single level of recursion.

of the patrol in the (newly) selected location. One can see that the constraint that generated this new node is the same as the constraint that was responsible for this decision in the previously computed world. Similarly, two other constraints are used with new arguments that make them come to different conclusions. These nodes are added to the existing graph and the new edges are drawn; edges that are no longer valid are changed to dashed. The resulting graph shows both the initial world and the newly-computed world (Figure 6). There is still some chance that Violence will occur in Atown (right side of the graph; compare the two nodes in the middle of the column). The patrol is no longer in Ctown; tag91, which placed the patrol in Ctown in the initial world (Figure 2), is not part of the new world's graph (dashed lines only). Therefore, the new world (after the decision) has a slightly increased chance that Violence will occur in Ctown (node at bottom right versus node at top right).

Finally, on this more complex world, we can illustrate a useful case for the second interaction, which was

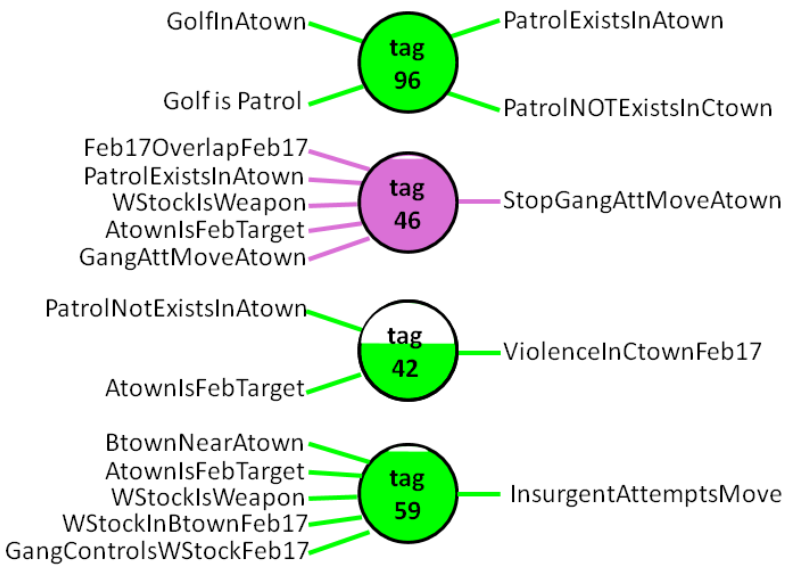

Figure 5. Four new constraint nodes (three green for Tactics, one lavender for Behaviors) are introduced to the world computed after the user changes a decision that was indicated in the initial world. Their consequents and antecedents lead to different (new) Outcomes (red nodes, not shown here; see Figure 6) in the most likely world, given the change to the decision set.

specifically (and independently) suggested by a domain expert. We use a recursive search through the graph from an outcome to identify decision points that contributed to that outcome. The nodes and edges are highlighted through increased line width; decision nodes that are not involved are faded. See Figure 7 for an example.

One observation of these interactions is that the Graphviz tool does not have all the capabilities that we desire for our graphs. We perform a semi-automated post-processing of the resulting graph image in order to fix certain attributes. Nodes cannot be partly filled; this is a useful visual representation of uncertainty that we would like to use. Certain style attributes cannot be combined, such as the "bold" edges and filled nodes in the trace from result to evidence (Figure 7). Filling or (partly) unfilling nodes is relatively easy, but would not be necessary with a graphing tool customized to our drawing guidelines. Another, more difficult challenge to overcome is the change in layout that results from introducing new nodes. Ideally, nodes that were in a graph before an interaction would not move when bubbles pop up to explain the reasoning at nodes or when new nodes are introduced by new constraints or alternative decisions. These issues cannot be overcome without a customized graphing tool, which we leave as future work.

5. DISCUSSION

Our method, though still a prototype, certainly meets the first criterion of adequacy that we identified (cognitive basis); the use of Polyscheme gives us this in our implementation. To assess whether the visualizations and interactive operations combine to meet the second and third criteria, we showed early versions of the visualizations of the scenario (including the alternative decision) described in the previous section to domain experts in cognitive architectures. (We added constraints after these expert design reviews. This forced Polyscheme to explicitly reason about Outcomes that were true by default, which in turn enabled our visualization algorithm to place them in the graphs in Figures 3-7. The recommendations from the expert users were unrelated to these changes.)

Our first expert was quite familiar with Polyscheme. His first comment was that he liked that we were able to avoid displaying much of the "extraneous" information that Polyscheme needs, but to humans is obvious – e.g. 17 February is within the month of February. Obvious as this is to humans, Polyscheme's processing of time requires explicit constraints developing relationships between days and months, and sequencing. A higher-level example is the proximity of two locations. To our end users (whom we assume will be familiar with the terrain), it will likely be common knowledge (as in our example) that Atown is near Btown. But Polyscheme currently requires a constraint expressing the symmetric property of this relationship. This raises some deeper issues, however.

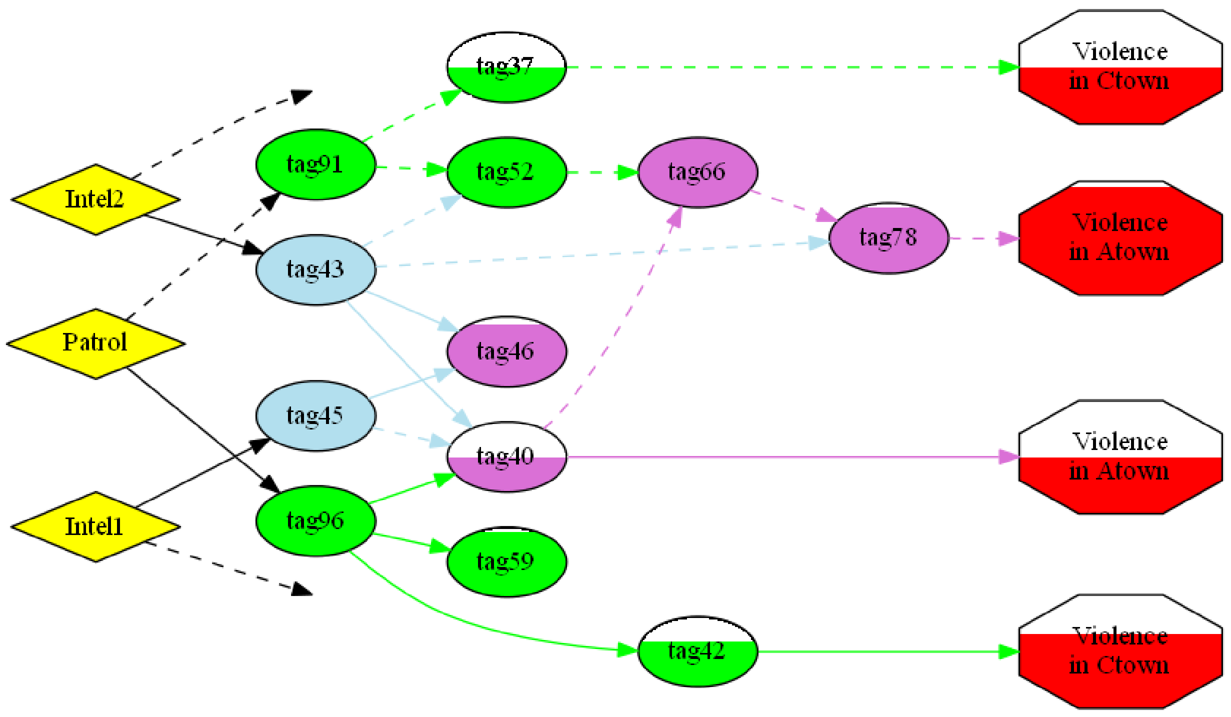

Figure 6. In the world that occurs with the new decision (assigning the patrol to Atown), there is no longer a high likelihood to Violence in Atown. However, there is a likelihood of Violence in Ctown, since the patrol is no longer there. New nodes in the graph appear generally in the bottom half. Edges that belong only to the first world (before the new decision) are dashed, and nodes that belong only to the first world have no border.

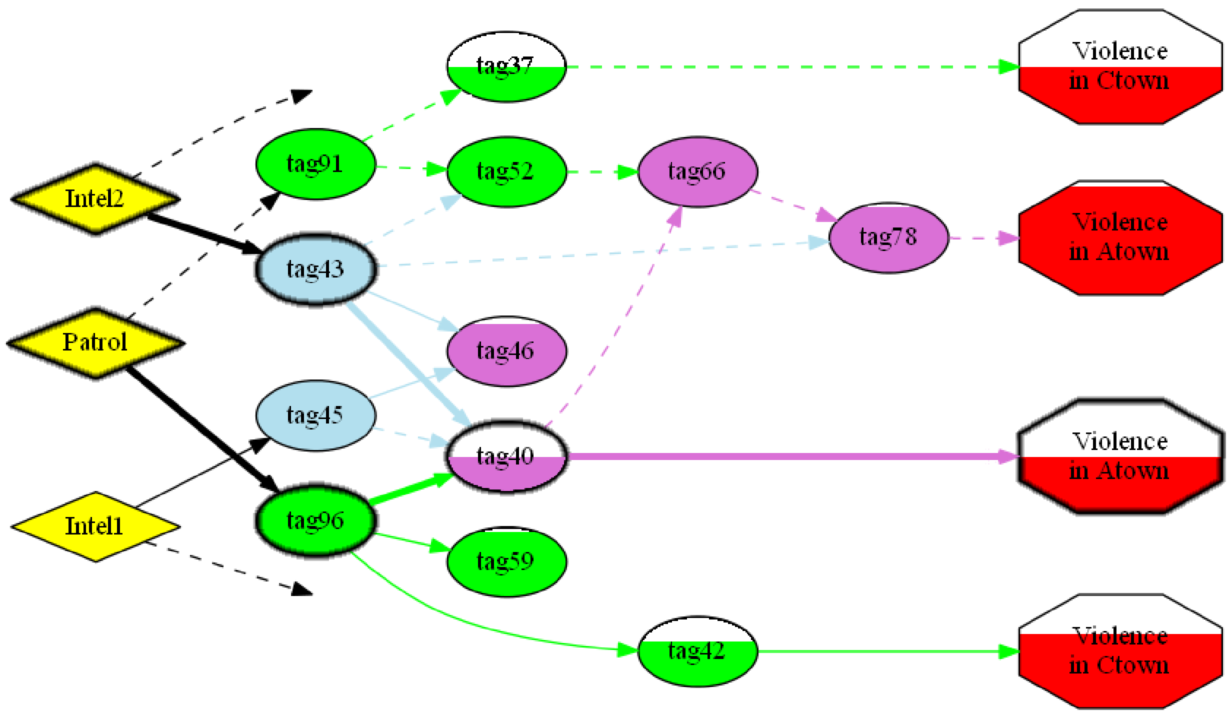

Figure 7. The recursive trace operation enables the user to select an Outcome and highlight the trace through the graph back to decision points that contributed to that Outcome. In this simple example, two decisions make a contribution to the level of Violence the model estimates for Atown.

If terrain issues could be transitory during emergency scenarios, such as whether a road has been flooded and rendered impassable by a storm, then there may need to be awareness of terrain status pushed to users who are familiar with an area of operations. Still the high-level views that we provided without unnecessary details were viewed favorably.

Both of our experts were enthusiastic about the possibility of being able to view alternate scenarios. This was compared favorably with other cognitive architectures. ACT-R can display the global view of its system model at the end of its computations. This maps back to buffers that predict neural activations (with temporal sequencing). But this model is at a much lower level in the reasoning process, so the comparison is strained; this was viewed as a positive contribution that our system can make to the field. One could argue that our system needs more knowledge, since ACT-R has a wealth of neural models, whereas we must program Polyscheme with constraints. However, we get dependency trees and have far less-constrained worlds, while ACT-R gives a graph of the firing of neurons, but not really a causal chain.

One of our experts suggested that interaction would be critical. Several specific suggestions were made (before we showed the existing interaction operations). First, as a minor display change, users could specify short codes that could replace the numerical instance IDs that we currently display in the nodes. This may enable users to create mnemonics for actions or states that would assist with comprehension of the graph. Second, the recursive trace from outcomes to decisions was suggested as a useful interaction. (We then presented our design for this to the expert.) Finally, the expert recommended a tool that would highlight all the alternate worlds where a certain outcome occurs. Then the user could examine those worlds for a pattern of decisions, to better understand the consequences of certain decisions. This alludes to the issue of how decisions may interact with each other in complex worlds where we ask Polyscheme to reason through uncertain information. We believe this will be a useful tool as we apply our implementation to more complex scenarios.

The final comment was a favorable view on our presentation of uncertainty at each node. This was considered reminiscent of Slate, which has strength factors in a $[-5, 5]$ range. Slate, which does not explicitly perform probabilistic inference, propagates the weakest strength factor among its antecedents to its inferred consequents. The strength of an inference is visually represented as text appended to the description of the node; for example, the strength 1 is described by the text "Probable" while strength 2 by "Beyond Reasonable Doubt." Our visualization of uncertainty, in which the amount of fill of a node indicates the probability, was favored.

One comment that we add based on our experiences is that this system as it exists now requires a fairly extensive programming effort in Polyscheme's modeling language. This is clearly not a task for the end-user, who is unlikely to be an expert in cognitive modeling in general or Polyscheme in particular. A tool to help the end user write constraints for the world would make the algorithm itself much more accessible.

Still, we believe that the comments from the experts in cognitive architectures support our assertion that we are moving towards meeting all three conditions of adequacy. Our methods distill the knowledge down to the non-obvious information and how it gets combined to reach conclusions about likelihood of outcomes. But we think that the graph representation of this reasoning that we have implemented is currently easily understood, and even with the enhancements suggested above, will remain comprehensible, while still enabling access to oftentimes complex reasoning of scenarios.

ACKNOWLEDGMENTS

The authors wish to thank Nicholas Cassimatis, Jonathan Decker, Bruce Gudmundsson, Sangeet Khemlani, and Unmesh Kurup. This work was supported by the NRL Base Program.

REFERENCES

[1] Anderson, J. R., [*How Can the Human Mind Occur in the Physical Universe?*], Cognitive Models and Architectures, Oxford University Press (2007).

[2] Laird, J. E., Newell, A., and Rosenbloom, P. S., "Soar: An architecture for general intelligence," *Artificial Intelligence* **33**, 1–64 (Jan. 1987).

[3] Cassimatis, N. L., "A cognitive substrate for achieving human-level intelligence," *AI Magazine* **27**(2), 45–56 (2006).

[4] Lakoff, G. and Johnson, M., [*Philosophy in the Flesh: The Embodied Mind and Its Challenge to Western Thought*], Chicago: The University of Chicago Press (1999).

[5] Pearl, J., [*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*], Morgan Kaufmann (1988).

[6] Richardson, D. C. and Spivey, M. J., "Spatial representations activated during real-time comprehension of verbs," *Cognitive Science* **27**, 767–780 (2003).

[7] Jackendoff, R., [*Semantic Structures*], The MIT Press (1990).

[8] Cree, G. S. and McRae, K., "Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns)," *Journal of Experimental Psychology: General* **132**(2), 163–201 (2003).

[9] Cassimatis, N. L., "Integrating cognitive models based on different computational methods," in [*Proceedings of the XVII Annual Conference of the Cognitive Science Society*], Bara, B. G., Barsalou, L., and Bucciarelli, M., eds., 402–407, Lawrence Erlbaum Associates (July 2005).

[10] Allwein, G. and Barwise, J., eds., [*Logical Reasoning with Diagrams*], Studies in Logic and Computation, Oxford University Press (1996).

[11] Gooding, D. C., "Visualizing scientific inference," *Topics in Cognitive Science* **2**, 15–35 (Jan. 2010).

[12] Michalski, R. S., "Graphical minimization of normal expressions of logic functions using tables of the veitch-karnaugh type," *Journal of the Institute of Automatic Control* **52** (1978).

[13] Bringsjord, S., Taylor, J., Shilliday, A., Clark, M., and Arkoudas, K., "Slate: An argument-centered intelligent assistant to human reasoners," in [*Proceedings of the 8th International Workshop on Computational Models of Natural Argument*], 1–10 (July 2008).

[14] Boley, H., "RuleML/Grailog: The rule metalogic visualized with generalized graphs," in [*PhiloWeb 2011*], (Oct. 2011).

[15] Kontopoulos, E., Bassiliades, N., and Antoniou, G., "Visualizing semantic web proofs of defeasible logic in the DR-DEVICE system," *Knowledge-Based Systems* **24**, 406–419 (2011).

[16] Sharara, H., Sopan, A., Namata, G., Getoor, L., and Singh, L., "G-Pare*: A visual analytic tool for comparative analysis of uncertain graphs," in [*IEEE Symposium on Visual Analytics Science and Technology*], 59–68 (Oct. 2012).

[17] Burch, M., Heinrich, J., Konevtsova, N., Höferlin, M., and Weiskopf, D., "Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study," *IEEE Transactions on Visualization and Computer Graphics* **17**, 2440–2448 (Dec. 2011).

[18] Bremm, S., von Landesberger, T., Heß, M., Schreck, T., Weil, P., and Hamacher, K., "Interactive visual comparison of multiple trees," in [*IEEE Symposium on Visual Analytics Science and Technology*], 29–38 (Oct. 2012).

[19] Gómez, J. A. G., Buck-Coleman, A., Plaisant, C., and Shneiderman, B., "TreeVersity: Comparing tree structures by topology and node's attributes differences," in [*IEEE Symposium on Visual Analytics Science and Technology*], 273–274 (Oct. 2012).

[20] Trafton, J. G., Cassimatis, N. L., Bugajska, M. D., Brock, D. P., Mintz, F. E., and Schultz, A. C., "Enabling effective human-robot interaction using perspective-taking in robots," *IEEE Transactions on Systems, Man, and Cybernetics* **35**, 460–470 (2005).

[21] Cassimatis, N. L., Murugesan, A., and Bignoli, P. G., "Inference with relational theories over infinite domains," in [*Proceedings of the Twenty-Second International FLAIRS Conference*], 21–26 (2009).

[22] Clancey, W. J., [*Extensions to Rules for Explanation and Tutoring*], ch. 29, 531–568, Addison-Wesley (1984).

[23] Chajewska, U. and Halpern, J. Y., "Defining explanation in probabilistic systems," in [*Proceedings of the Thirteenth Conference on Uncertainty in AI*], 62–71 (1997).

[24] Henrion, M. and Druzdzel, M. J., "Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning," *Uncertainty in Artificial Intelligence* **6**, 17–32 (1990).

[25] Ellson, J., Gansner, E., Koutsofios, L., North, S., and Woodhull, G., "Graphviz – open source graph drawing tools," in [*Graph Drawing (Lecture Notes in Computer Science)*], 483–484 (2001).